FIGURE 1

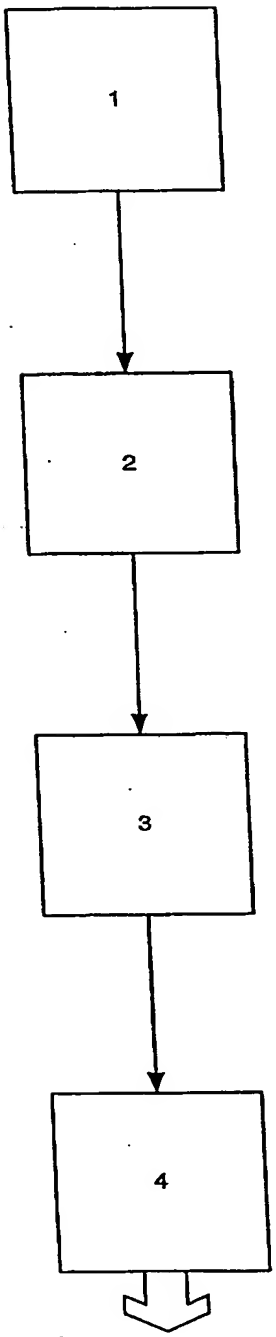

FIGURE 2

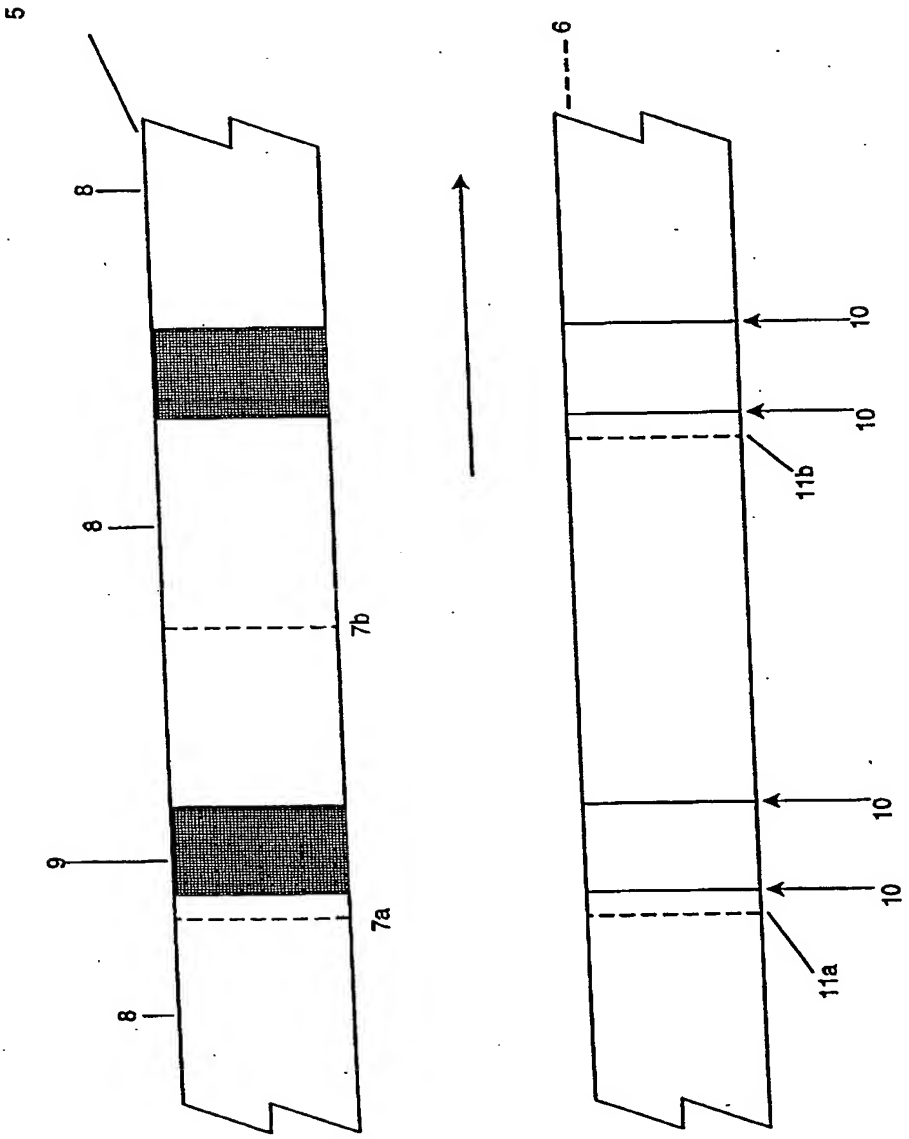

FIGURE 3a

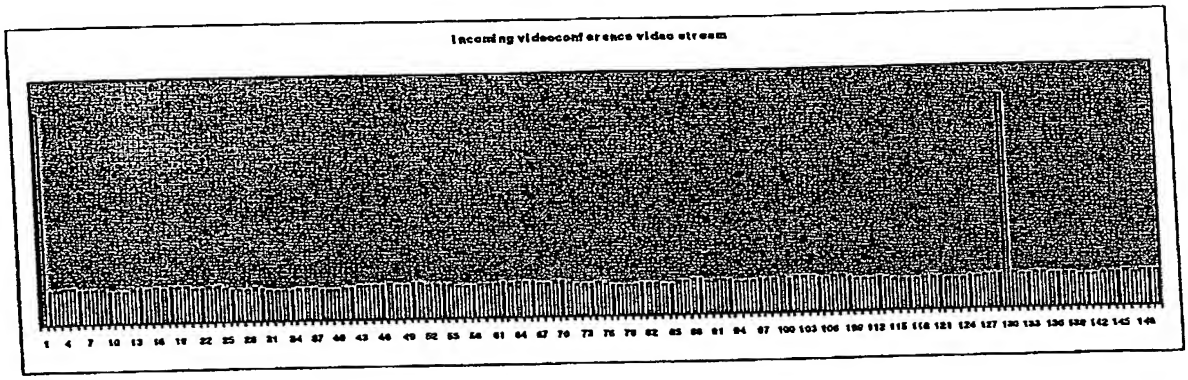

FIGURE 3b

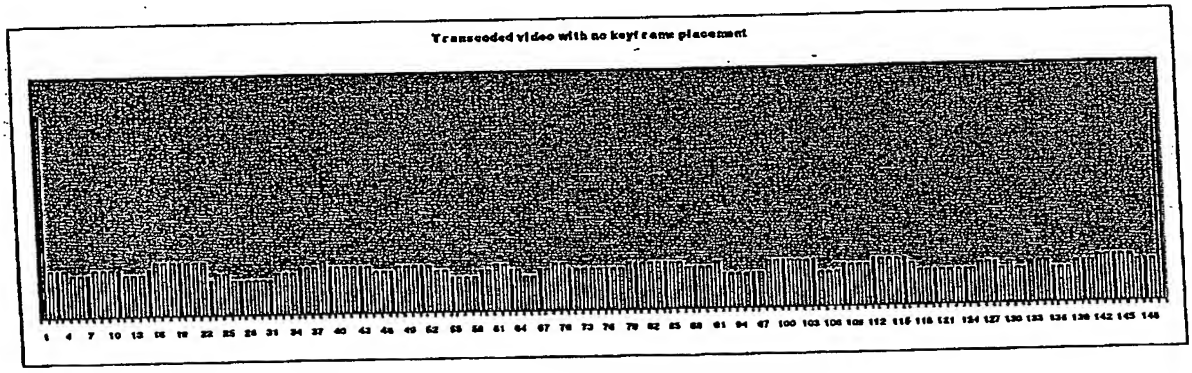

FIGURE 3c

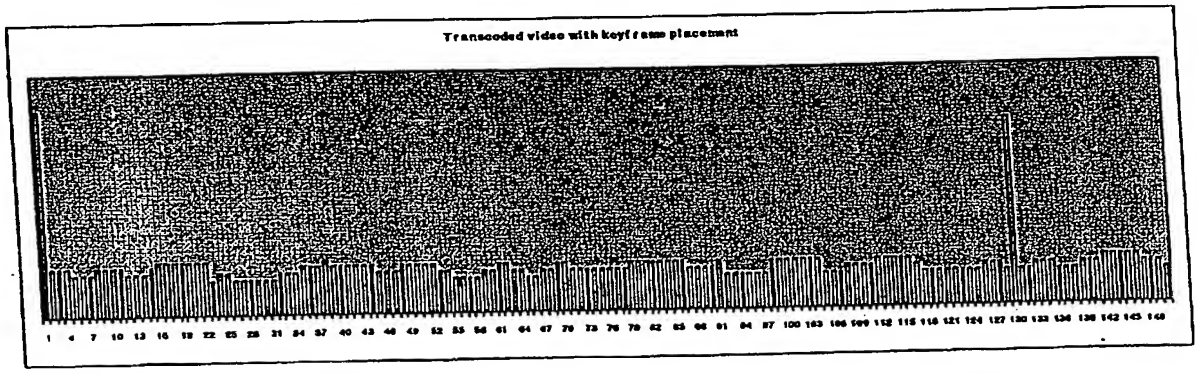

FIGURE 4

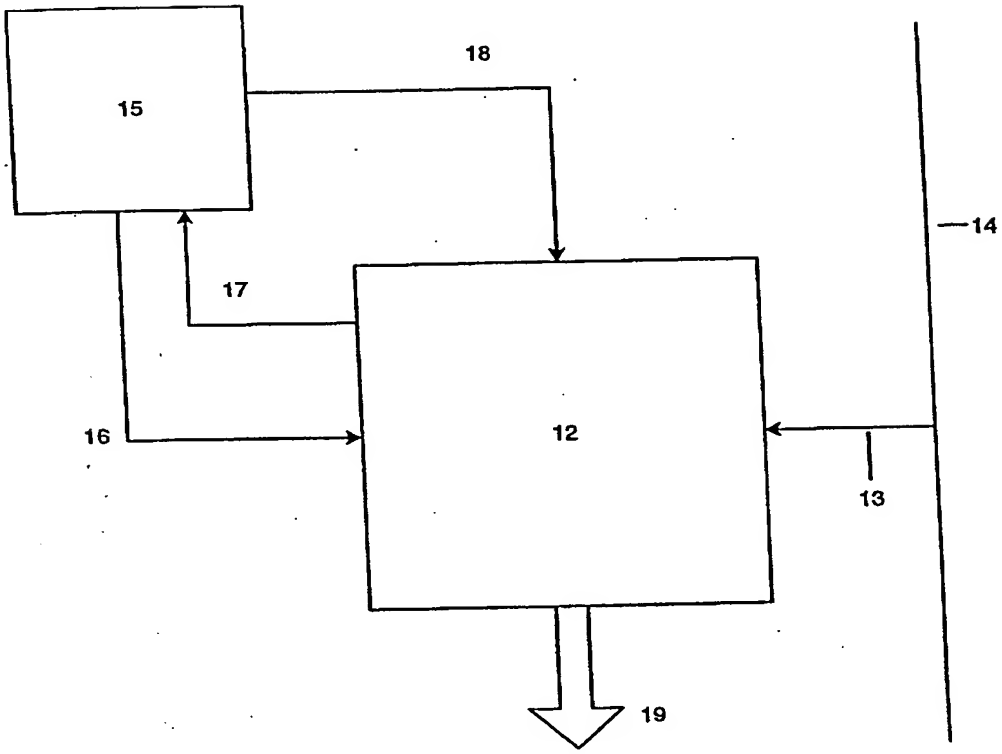

FIGURE 5a

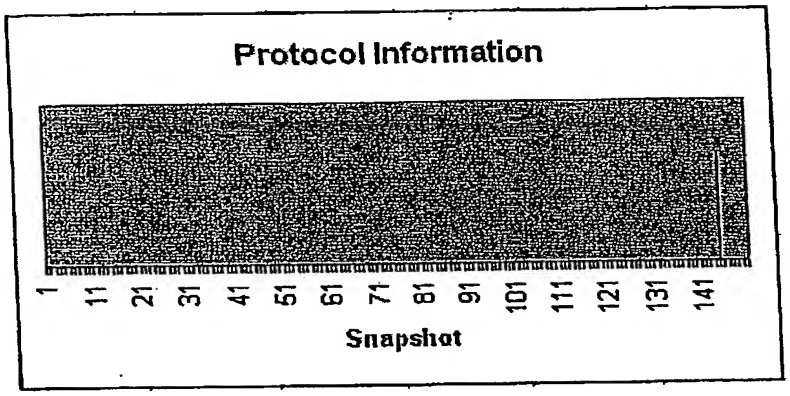

FIGURE 5b

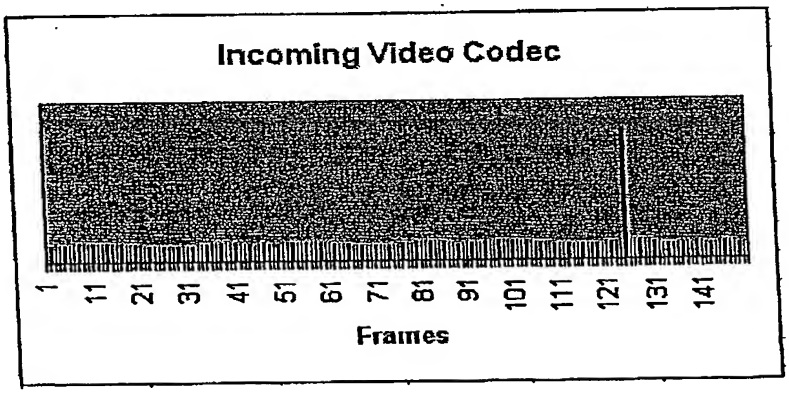

FIGURE 5c

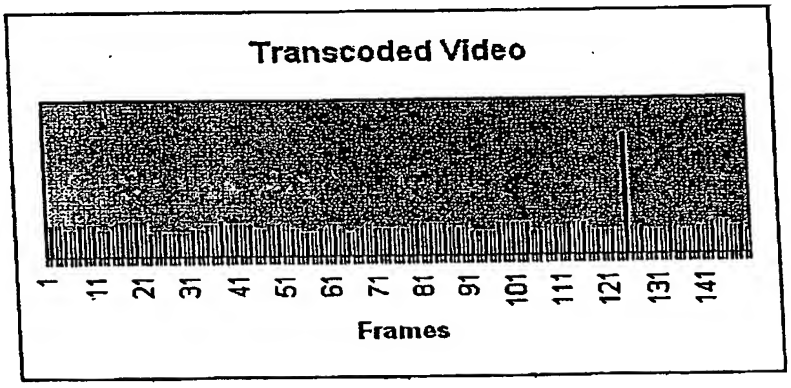

FIGURE 6a

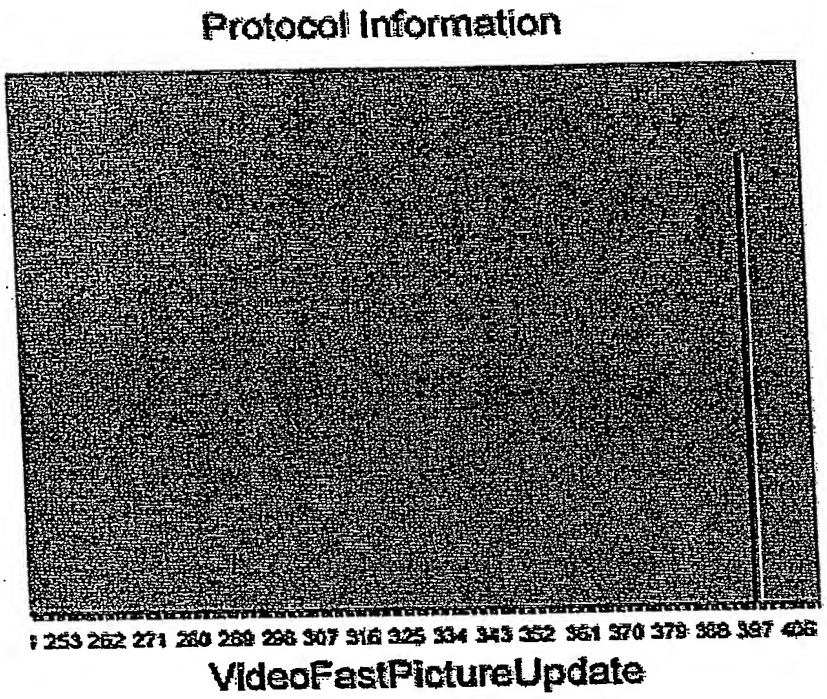

FIGURE 6b

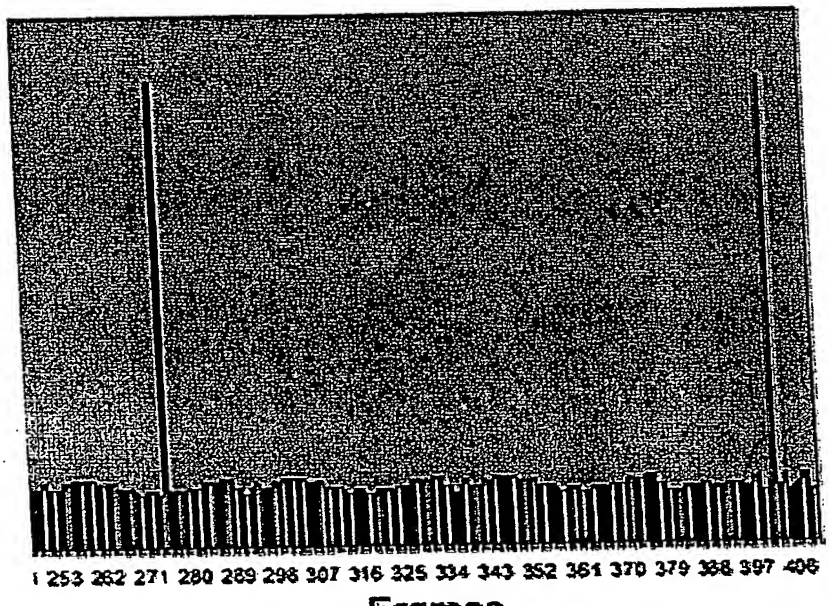

FIGURE 6c

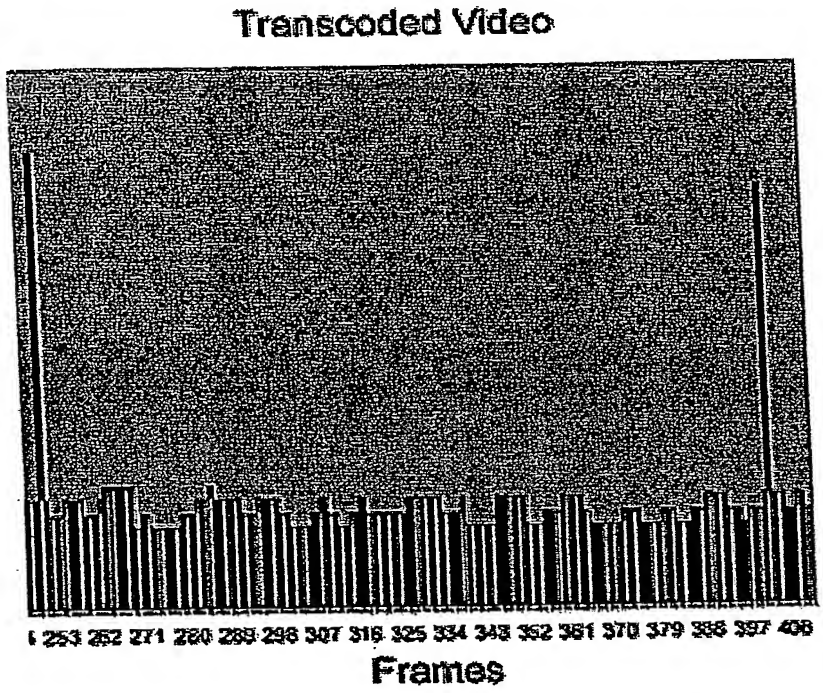

FIGURE 7

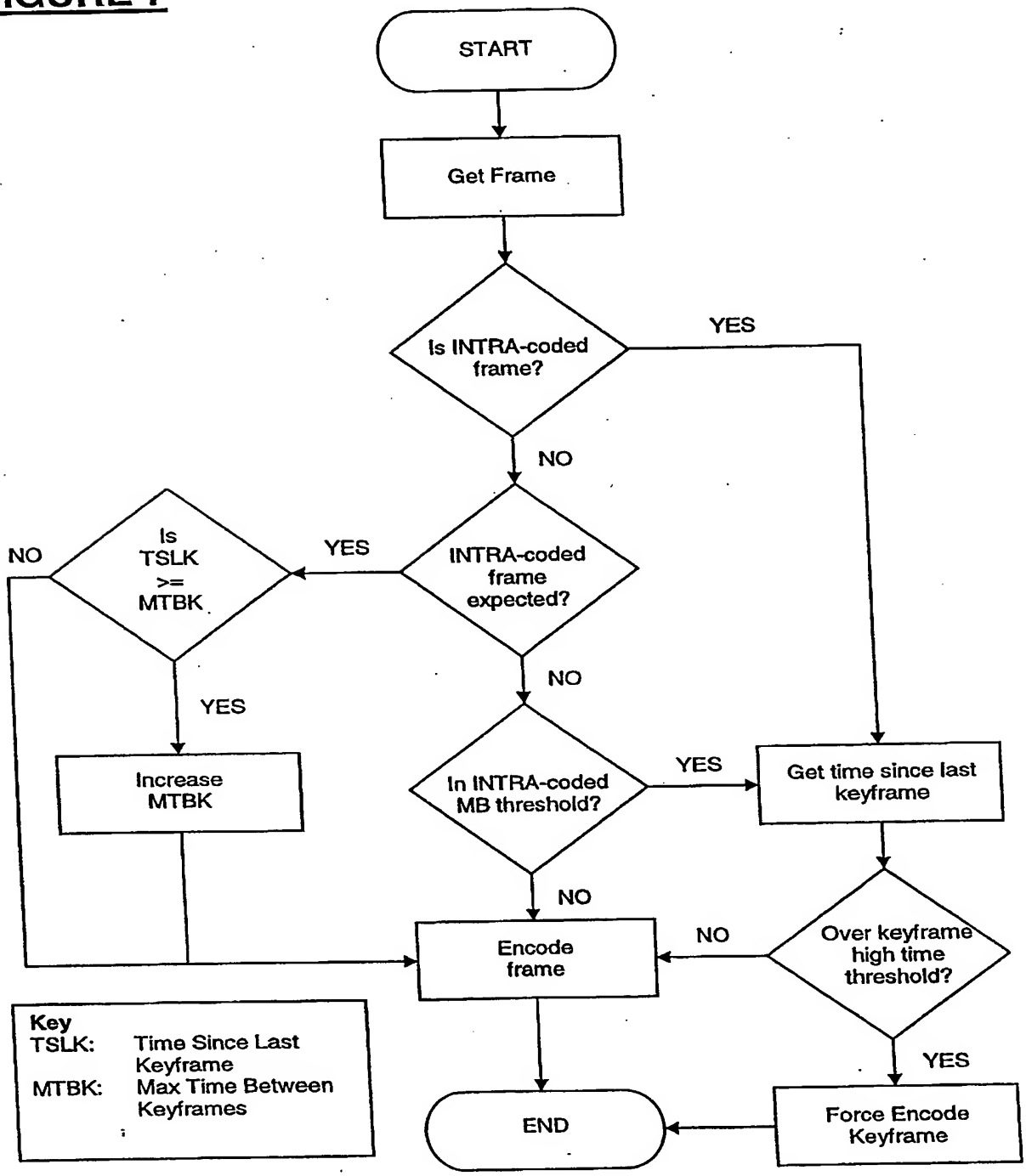

Table 1

```
cifMacroblocks = 396
qcifMacroblocks = 99
macroblock threshold = 0.85
max time between keyframes = 10sec

for (every received frame)

      get next frame
      get frame type
      get (macroblock count) for frame

      if frame type == CIF
            if (macroblock count) == cifMacroblocks then frame is INTRA-coded
            (macroblock threshold count) = cifMacroblocks * macroblock
threshold
      else if frame type == QCIF
            if (macroblock count) == qcifMacroblocks then frame is INTRA-coded
            (macroblock threshold count) = qcifMacroblocks * macroblock
threshold
      end if

      if frame is INTRA-coded
            if (Check Force Keyframe)
                  force encode keyframe
            else
                  standard encode frame
            end if

      else if INTRA-coded frame expected
            if (time since last keyframe) >= (max time between keyframes)
                  increase (max time between keyframes)
            else
                  standard encode frame

      else if INTRA-coded MB threshold
            if (macroblock count) >= (macroblock threshold count)
                  if (Check Force Keyframe)
                        force encode keyframe
                  else
                        standard encode frame
                  end if
            else
                  standard encode frame

      end if

end for

Check Force Keyframe
BEGIN

      get (time since last keyframe)
      get (keyframe threshold)
      keyframeCheck = (max time between keyframes) * (keyframe threshold)
      if (time since last keyframe) >= keyframeCheck
            return true
      else
            return false
      end if
END
```

Title: AUDIO VISUAL MEDIA ENCODING SYSTEM
Inventor(s): Craig George COCKERTON
Atty. Docket No. 1519-053

FIGURE 8

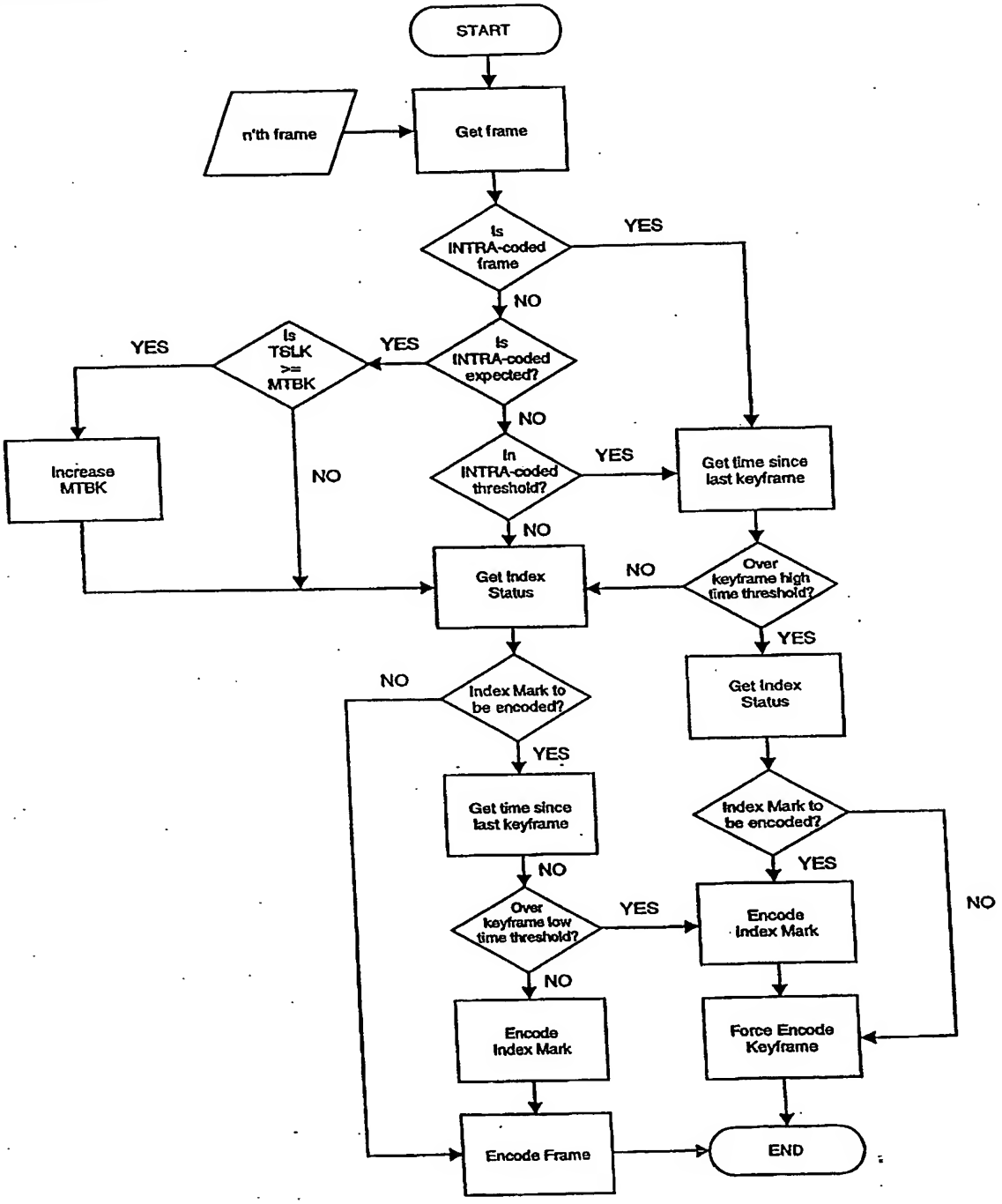

Table 2

```
cifMacroblocks = 396
qcifMacroblocks = 99
macroblock threshold = 0.85
max time between keyframes = 10sec

for (every received frame)

      get next frame
      get frame type
      get INTRA-coded (macroblock count) for frame

      if frame type == CIF
            if (macroblock count) == cifMacroblocks then frame is INTRA-coded
            (macroblock threshold count) = cifMacroblocks * macroblock
threshold
      else if frame type == QCIF
            if (macroblock count) == qcifMacroblocks then frame is INTRA-coded
            (macroblock threshold count) = qcifMacroblocks * macroblock
threshold
      end if

      if frame is INTRA-coded
            if (Force Keyframe Threshold)
                  go to Forced Keyframe Index
            else
                  go to Standard Encode Index
            end if
      else
            if INTRA-coded frame expected
                  if (time since last keyframe) >= (max time between
keyframes)
                        increase (max time between keyframes) by
                              (max time to live)
                        store index data

                        go to Standard Keyframe Index
                  end if
            else
                  if INTRA-coded MB threshold
                        if (Force Keyframe Threshold)
                              go to Forced Keyframe Index
                        else
                              go to Standard Keyframe Index
                        end if
                  else
                        go to Standard Keyframe Index
                  end if
            end if
      end if
end for
```

Table 2 continued

```
Forced Keyframe Index

BEGIN

      if (Get Index Status)
            encode index mark
      end if

      force encode keyframe
      return

END

Standard Encode Index

BEGIN

      if (Get Index Status)
            encode index mark

            if (Index Keyframe Threshold)
                  force encode keyframe
            else
                  standard encode keyframe
            end if

      else
            standard encode keyframe
      end if

END

Index Keyframe Threshold

BEGIN

      get (time since last keyframe)
      get (keyframe index threshold)
      keyframeCheck = (max time between keyframes) * (keyframe index threshold)
      if (time since last keyframe) >= keyframeCheck
            return true
      else
            return false
      end if

END

Force Keyframe Threshold

BEGIN

      get (time since last keyframe)
      get (keyframe threshold)
      keyframeCheck = (max time between keyframes) * (keyframe threshold)
      if (time since last keyframe) >= keyframeCheck
            return true
      else
            return false
      end if

END
```

Title: AUDIO VISUAL MEDIA ENCODING SYSTEM
Inventor(s): Craig George COCKERTON
Atty. Docket No. 1519-053

FIGURE 9

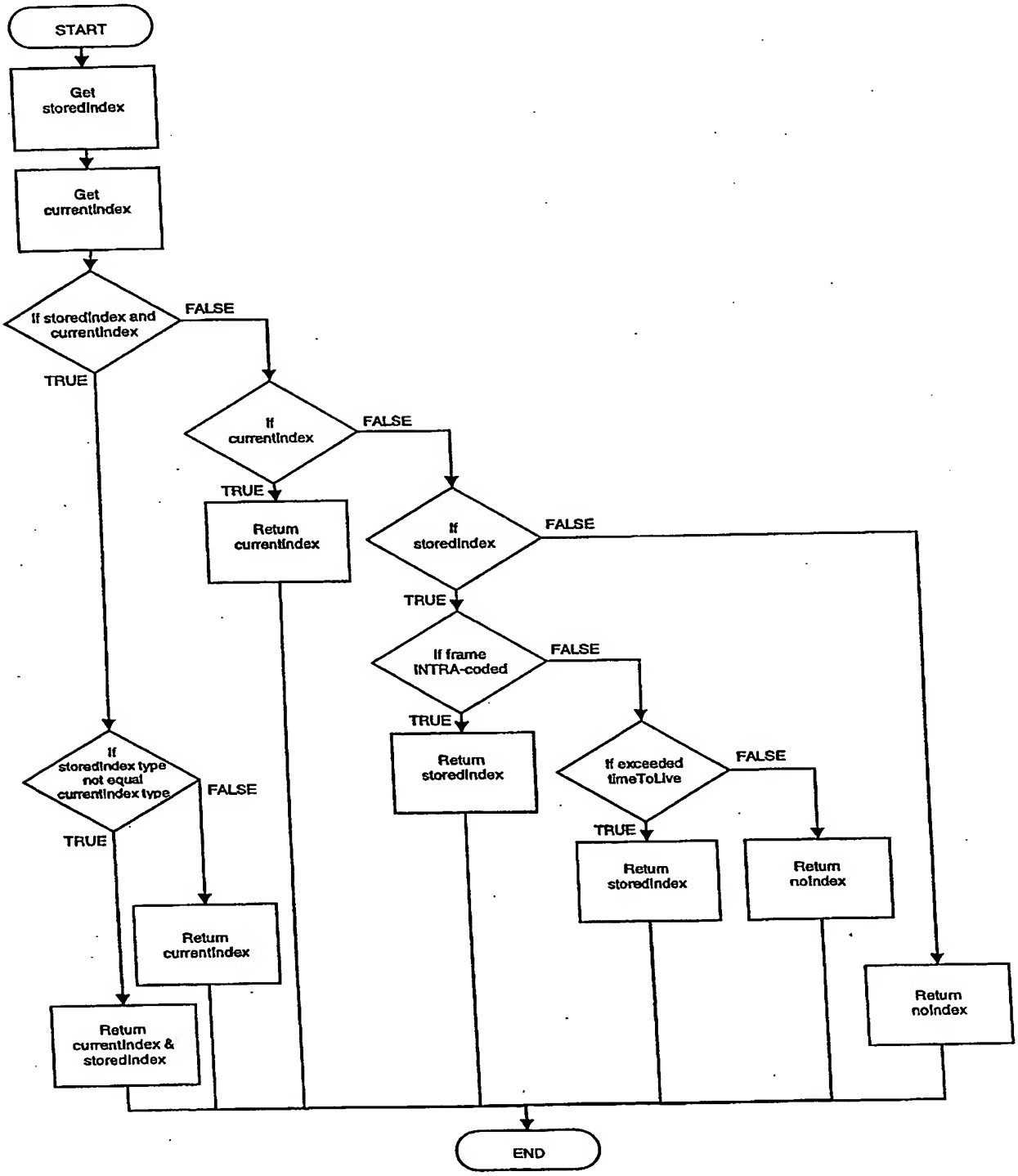

Table 3

```
Get Index Status

BEGIN

      get stored index
      get current index

      if (stored index) and (current index)

            if (stored index type) != (current index type)
                  return (stored index) and (current index)
            else
                  return (current index)
            end if

       else if (current index)
             return (current index)

       else if (stored index)

             if frame is INTRA-coded
                   return (stored index)
             else if (stored index) exceeded time to live
                   return (stored index)
             else
                   return (no index)
             end if

        else
              return (no index)
        end if

 END
```

FIGURE 10

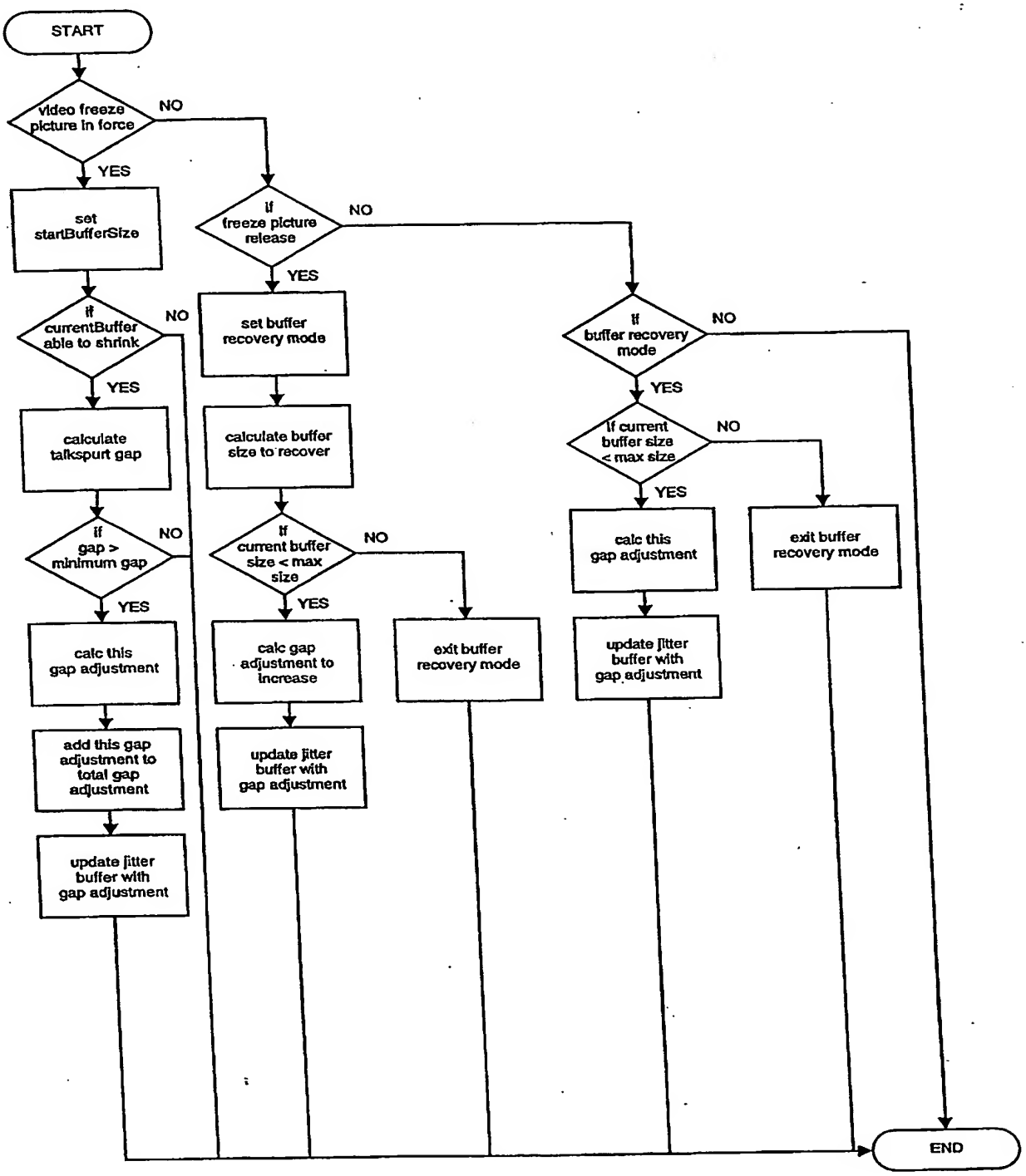
START
video freeze picture in force
NO
YES
set startBufferSize
if freeze picture release
NO
YES
if currentBuffer able to shrink
NO
YES
set buffer recovery mode
if buffer recovery mode
NO
YES
calculate talkspurt gap
calculate buffer size to recover
if current buffer size < max size
NO
YES
if gap > minimum gap
NO
YES
if current buffer size < max size
NO
YES
calc this gap adjustment
exit buffer recovery mode
calc this gap adjustment
exit buffer recovery mode
update jitter buffer with gap adjustment
calc gap adjustment to increase
update jitter buffer with gap adjustment
add this gap adjustment to total gap adjustment
update jitter buffer with gap adjustment
END

Table 4

```
Jitter Buffer Adjust

BEGIN

      if (video freeze picture) in force

            get currentBufferSize
            if startBufferSize not set
                  set startBufferSize to currentBufferSize
            end if

            if currentBufferSize > (minBufferSize * bufferAdjustRatio)

                  calculate gap between current and next talkspurts

                  if (gap > minimumGap)
                        adjustGap = (gap - minimumGap) * adjustLevel

                        if adjustGap > maxAdjustGap
                              adjustGap = maxAdjustGap
                        end if

                        tell jitterBuffer adjustGap
                        // adjusting the gap between talkspurts to playout
                        // the packets earlier will cause the buffer to
                        // drain quicker.
                  end if

            end if

      else if (freeze picture release)
      // actual freeze picture release could be received in multiple video
      // packets but we only want to do this stuff once so this needs to be
      // controlled by calling function

            set bufferMadeUpSoFar to 0
            set bufferRecovery true
            get currentBufferSize
            set endBufferSize to currentBufferSize

            set bufferToMakeUp = startBufferSize - endBufferSize
            // we need to makeup the buffer size that we drained by
            // playing out packets quicker than normal

            if (currentBufferSize < maxBufferSize)

                  adjustGap = increaseGap * (1+(1-adjustLevel))

                  if (adjustGap + bufferMadeUpSoFar) > bufferToMakeUp

                        adjustGap = bufferToMakeUp - bufferMadeUpSoFar
                        bufferRecovery = false

                  end if

                  increase bufferMadeUpSoFar by adjustGap
                  tell jitterBuffer adjustGap
```

Table 4 continued

```
            else

                  bufferRecovery = false

            end if

      else if (bufferRecovery)
      // we've done the freeze picture release to start the recovery now
it's
      // time to complete it by increasing the talkspurt gaps until we
recover
      // from the draining of the buffer during the video freeze picture.

            if (currentBufferSize < maxBufferSize)
                  adjustGap = increaseGap * (1+(1-adjustLevel))

                  if (adjustGap + bufferMadeUpSoFar) > bufferToMakeUp

                        adjustGap = bufferToMakeUp - bufferMadeUpSoFar
                        bufferRecovery = false

                  end if

                  increase bufferMadeUpSoFar by adjustGap
                  tell jitterBuffer adjustGap

            else

                  bufferRecovery = false

            end if

      end if

END
```